REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN
CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202271593, 4 Oktober 2022
Pencipta		
Nama	:	**Okta Lestari, Asniar Aliyu, S.T., M.Eng. dkk**
Alamat	:	Mejing Kidul RT 05/08, Ambarketawang, Gamping, Sleman, Yogyakarta, Sleman, DI YOGYAKARTA, 55295
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**LPPMI ITNY**
Alamat	:	Jl. Babarsari, Tambak Bayan, Caturtunggal, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta , Sleman, DI YOGYAKARTA, 55281
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**PROGRAM MIKROKONTROLER UNTUK PERANCANGAN SISTEM PENETAPAN TINGKAT KEPADATAN MULTI RUANG MENGGUNAKAN ESP32**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	10 Agustus 2022, di Yogyakarta
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor pencatatan	:	000387334

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n Menteri Hukum dan Hak Asasi Manusia
Direktur Jenderal Kekayaan Intelektual
u.b.
Direktur Hak Cipta dan Desain Industri

Anggoro Dasananto
NIP.196412081991031002

Disclaimer:
Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Okta Lestari	Mejing Kidul RT 05/08, Ambarketawang, Gamping, Sleman, Yogyakarta
2	Asniar Aliyu, S.T., M.Eng.	Pogung Kidul, RT 01/49, Sinduadi, Mlati, Sleman, Yogyakarta
3	Arif Basuki, S.T., M.T.	Basen RT13/04, Purbayan, Kotagede, Indonesia

DOKUMEN
HAK KEKAYAAN INTELEKTUAL KARYA CIPTA

PROGRAM MIKROKONTROLER UNTUK PERANCANGAN SISTEM PENETAPAN TINGKAT KEPADATAN MULTI RUANG MENGGUNAKAN ESP32

Pencipta:

Okta Lestari
Asniar Aliyu, ST. M.Eng
Arif Basuki, ST. MT

INSTITUT TEKNOLOGI NASIONAL YOGYAKARTA

2022

PROGRAM MIKROKONTROLER UNTUK PERANCANGAN SISTEM PENETAPAN TINGKAT KEPADATAN MULTI RUANG MENGGUNAKAN ESP32

Perancangan sistem penetapan tingkat kepadatan multi ruang menggunakan ESP32 ini merupakan sebuah prototipe berbasis IoT yang didesain untuk dapat mendeteksi dan menghitung objek yang melewati pintu ketika akan masuk ke dalam suatu ruangan dan ketika jumlah orang dalam ruangan tersebut telah terpenuhi pintu akan terkunci secara otomatis serta datanya dapat dimonitoring menggunakan aplikasi Blynk Iot dan Blynk Console. Perancangan sistem penetapan tingkat kepadatan multi ruang menggunakan ESP32 terdiri dua ruangan, jadi dalam prototipe ini menggunakan dua buah sensor proximity E18-D80NK yang digunakan untuk mendeteksi keberadaaan objeknya, dua buah mikrokontroler dengan jenis ESP32 yang digunakan sebagai kontroler utamanya, dua buah solenoid *doorlock* yang digunakan untuk mengunci pintu secara otomatis, dua buah *buck converter* DC-DC yang digunakan untuk penurun tegangan agar ESP32 dan solenoid *doorlock* dapat berfungsi, dan 2 buah LCDI2C yang digunakan untuk menampilkan data informasinya.

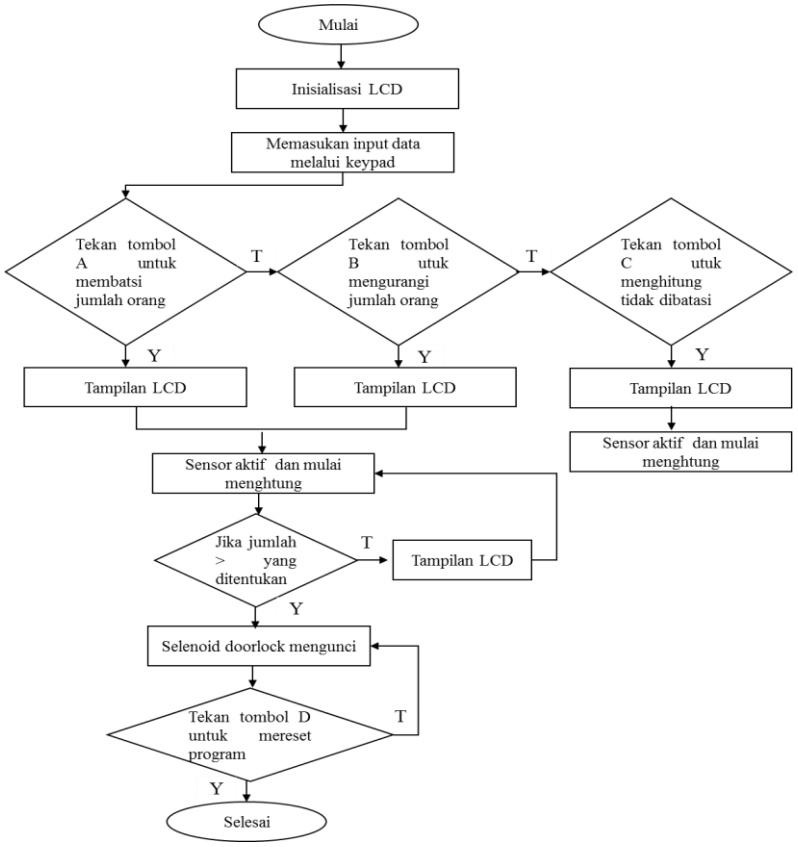
1. Prinsip Kerja

Prinsip kerja dari sistem penetapan tingkat kepadatan multi ruang menggunakan ESP32 ini ialah ketika sensor proximity E18-D80NK pada pintu masuk ON lalu menekan tombol A untuk memasukan data jumlah orang yang diinginkan setelah itu sensor proximity E18-D80NK akan mulai proses perhitungan. Dan data jumlah orang ini akan tertampil pada layar LCD. Proses

berikutnya adalah pengecekan apakah data jumlah orang melebihi data kapasitas maksimal jumlah orang dalam ruangan atau tidak. Jika jumlah orang melebihi kapasitas maksimal maka selenoid *doorlock* akan mengunci (ON), jika tidak maka selenoid *doorlock* OFF, jika ingin melakukan proses perhitungan lagi dengan menekan tombol D untuk reset. Dan data ini nantinya bisa dilihat secara realtime menggunakan aplikasi Blynk IoT dan Blynk Console.

2. Algoritma Pemrograman

Perancangan sistem penetapan tingkat kepadatan multi ruang menggunakan ESP32 ini memiliki algoritma pemrograman yang ditunjukkan dengan diagram-alir pada gambar berikut.



Gambar 1. Diagram-alir algoritma pemrograman

3. Listing Program Perancangan Sistem Penetapan Tingkat Kepadatan Multi Ruang Menggunakan ESP32

```
// Template ID, Device Name and Auth Token are provided by the Blynk.Cloud
// See the Device Info tab, or Template settings
/*
#define BLYNK_TEMPLATE_ID "TMPLdmHOSooB"
#define BLYNK_DEVICE_NAME "Test Alat"
#define BLYNK_AUTH_TOKEN "_bj8SjfXcgrdHF67eVQhAeNvaPi4Qycz"
*/
#define BLYNK_TEMPLATE_ID "TMPL2PdlB3z5"
#define BLYNK_DEVICE_NAME "alat 1"
#define BLYNK_AUTH_TOKEN "pSI27xAVLkRk_VMVxiVe_JxVQIbo3GKV"
// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
char auth[] = BLYNK_AUTH_TOKEN;
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Galaxy A016394";
char pass[] = "utcy9169";
BlynkTimer timer;
#include <WiFiUdp.h>
#include <NTPClient.h>
// Variables to save date and time
String formattedDate;
String dayStamp;
String timeStamp;
String weekDays[7] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
String months[12] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};
// Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);
#include "Keypad.h"
const byte ROWS = 4;        // four rows
const byte COLS = 4;        // four columns
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
```

```
};
byte rowPins[ROWS] = {27, 14, 12, 13};
byte colPins[COLS] = {19, 18, 5, 17};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
#include "Wire.h"
#include "LiquidCrystal_I2C.h"
LiquidCrystal_I2C lcd(0x27, 16, 2);
const int sensorPin = 23;   // the number of the Sensor pin
const int ledGreen  = 4;    // the number of the LED Green pin
const int ledRed    = 15;   // the number of the LED Red pin
const int relay     = 26;   // the number of the Relay pin
int sensorState = 0;         // variable for reading the sensor status
int countValue  = 0;
int val = 0;
boolean prestate;
boolean counting;
boolean counts;
boolean doorstate;
char key;
String StringVal;
// This function is called every time the Virtual Pin 0 state changes
BLYNK_WRITE(V0){
  // Set incoming value from pin V0 to a variable
  int value = param.asInt();
  // Update state
   if (value == 1){
    Blynk.virtualWrite(V0, 0);
    delay(3000);
    ESP.restart();
  }
  else {
    Blynk.virtualWrite(V0, 0);
  }
}
// This function sends Arduino's uptime every second to Virtual Pin 2.
void myTimerEvent(){
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V2, millis() / 1000);
}
void setup(){
  // Debug console
  Serial.begin(115200);
  pinMode(ledGreen,  OUTPUT);
  pinMode(ledRed,    OUTPUT);
  pinMode(relay,     OUTPUT);
```

```
pinMode(sensorPin, INPUT);
prestate  = false;
counting  = false;
counts    = false;
doorstate = false;
lcd.begin();      // initialize the LCD
lcd.backlight();  // Turn on the blacklight
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Okta Lestari");
lcd.setCursor(0,1);
lcd.print("3000190013");
delay(3000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Starting");
lcd.setCursor(0,1);
lcd.print("Blynk");
Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
// You can also specify server:
//Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
//Blynk.begin(auth, ssid, pass, IPAddress(192,168,190,87), 8080);
// Setup a function to be called every second
timer.setInterval(1000L, myTimerEvent);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Blynk");
lcd.setCursor(0,1);
lcd.print("Connected!");
delay(2000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Initialize");
lcd.setCursor(0,1);
lcd.print("NTPClient");
// Initialize a NTPClient to get time
timeClient.begin();
timeClient.setTimeOffset(25200);
// Set offset time in seconds to adjust for your timezone, for example:
// GMT +1 = 3600
// GMT +8 = 28800
// GMT -1 = -3600
// GMT 0 = 0
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Initialize NTP");
```

```
  lcd.setCursor(0,1);
  lcd.print("Success!");
  delay(2000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Pengunjung Yang");
  lcd.setCursor(0,1);
  lcd.print("Di Izinkan : ");
  digitalWrite(ledGreen, LOW);
  digitalWrite(ledRed, LOW);
  digitalWrite(relay, LOW);
}
void loop(){
  Blynk.run();
  timer.run();
    // You can inject your own code or combine it with other sketches.
  // Check other examples on how to communicate with Blynk. Remember
  // to avoid delay() function!
  runtimeProgram();
}
void runtimeProgram() {
  sensorState = digitalRead(sensorPin);
  sensorCount();
  noLimit()
  key = keypad.getKey();
  if (key){
    if (key >= '0' && key <='9') {
      counting = false;
      counts = false;
      enterAmount();
    }
    else if (key == 'A') {
      counting = true;
      counts = false;
      if (StringVal.length() > 0) {
        val = StringVal.toInt();
        if (val > 100) {
          invalidAmount();
          delay(3000);
          deleteMemory();
        }
        else {
          runProgram1();
        }
      }
    }
```

```
    else if (key == 'B') {
      if (countValue > 0) {
        countValue = countValue-1;
        if (counting == true) {
          digitalWrite(ledRed, LOW);
          digitalWrite(relay, LOW);
          runProgram1();
        }
        else if (counts == true) {
          runProgram2();
        }
      }
    }
    else if (key == 'C') {
      counting = false;
      counts = true;
      runProgram2();
    }
    else if (key == 'D') {
      counting = false;
      counts = false;
      deleteMemory();
    }
  }
}
void enterAmount() {
  StringVal += key;
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Pengunjung Yang");
  lcd.setCursor(0,1);
  lcd.print("Di Izinkan : ");
  lcd.setCursor(13,1);
  lcd.print(StringVal);
}
void invalidAmount() {
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Batas Maksimal");
  lcd.setCursor(0,1);
  lcd.print("100 Orang");
}
void deleteMemory() {
  StringVal = "";
  countValue = 0;
  Blynk.virtualWrite(V5, countValue);
```

```
   Blynk.virtualWrite(V1, val);
  digitalWrite(ledGreen, LOW);
  digitalWrite(ledRed, LOW);
  digitalWrite(relay, LOW);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Pengunjung Yang");
  lcd.setCursor(0,1);
  lcd.print("Di Izinkan : ");
}
void runProgram1() {
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Di Izinkan : ");
  lcd.setCursor(13,0);
  lcd.print(StringVal);
  lcd.setCursor(0,1);
  lcd.print("Total      : ");
  lcd.setCursor(13,1);
  lcd.print(countValue);
  Blynk.virtualWrite(V5, countValue);
  Blynk.virtualWrite(V1, val);
}
void runProgram2() {
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Total Pengunjung");
  lcd.setCursor(0,1);
  lcd.print(countValue);
  Blynk.virtualWrite(V5, countValue);
  Blynk.virtualWrite(V1, val);
}
void sensorCount() {
  if (sensorState == LOW && prestate == false && counting == true) {
    digitalWrite(ledGreen, HIGH);
    delay(100);
    digitalWrite(ledGreen, LOW);
    if (StringVal.length() > 0) {
      val = StringVal.toInt();
      if (val != countValue) {
        digitalWrite(ledRed, LOW);
        doorstate = true;
        countValue++;
        timeStamps();
      }
      else if (countValue >= val) {
```

```
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Total Pengunjung");
      lcd.setCursor(0,1);
      lcd.print("Terpenuhi!");
      delay(3000);
     }
     if (countValue == val) {
      digitalWrite(ledRed, HIGH);
      digitalWrite(relay, LOW);
     }
    }
    prestate = true;
    Serial.println(countValue);
    runProgram1();
  }
  else if (sensorState == HIGH && prestate == true && counting == true) {
    prestate = false;
  }
  if (doorstate == true) {
    doorOpen();
    doorstate = false;
  }
}
void noLimit() {
  if (sensorState == LOW && prestate == false && counts == true) {
    digitalWrite(ledGreen, HIGH);
    delay(100);
    digitalWrite(ledGreen, LOW);
    countValue++;
    timeStamps();
    prestate = true;
    Serial.println(countValue);
    runProgram2();
  }
  else if (sensorState == HIGH && prestate == true && counts == true) {
    prestate = false;
  }
}
void doorOpen() {\
  digitalWrite(relay, HIGH);
  delay(3000);
  digitalWrite(relay, LOW);
}
void timeStamps() {
  timeClient.update();
```

```
  unsigned long epochTime = timeClient.getEpochTime();
  struct tm *ptm = gmtime ((time_t *)&epochTime);
  int monthDay = ptm->tm_mday;
  int currentMonth = ptm->tm_mon+1;
  int currentYear = ptm->tm_year+1900;
  String weekDay = weekDays[timeClient.getDay()];
  String currentMonthName = months[currentMonth-1];
  String currentDate = String(weekDay) + ", " + String(monthDay) + " " +
String(currentMonthName) + " " + String(currentYear) + " ";
  Serial.print("Last access: ");
  Serial.print(currentDate);
  Serial.println(timeClient.getFormattedTime());
  Blynk.virtualWrite(V3, currentDate);
  Blynk.virtualWrite(V4, timeClient.getFormattedTime());
}
```